PHP on System i – It's Here!

(Part II)

By Shahar Mor

In the previous article we covered support of the PHP platform on System i. Although PHP is powerful, practical usage of PHP on System i requires integration capabilities to allow access of System i resources from the PHP code. Most System i applications are developed for business, and many customers will probably want to use PHP to develop Web-based user interfaces to existing business logic and databases. In order to do so, tools and extensions are needed.

Being aware of the customer requirements for PHP connectivity to System i resources, IBM ships Zendcore with two important extensions. One is the DB2 extension to allow database access (including stored procedure invocation) and the second is the PHP toolbox which allows several integration capabilities, including program call to existing programs (with parameters), sending and receiving messages from data queues (sequential or keyed), and more. This article covers some integration scenarios with some basic code samples to demonstrate PHP connectivity. The samples are not intended to teach HTML (What can you expect from an old green screen developer?) nor will you learn PHP from them, however they will introduce basic connectivity methods to get started.

Although completely out of the scope of this article, note that you will probably want to develop PHP on System i (or any other operating system) using the ZENDstudio. The green-screen environment may be used for simple file editing but it cannot be used to do serious complex PHP programming or even html authoring.

Sample Code 1:

```
<html>
<head><title>Work Active Job</title></head>
<body>
<p><h1>Work Active Job</h1>
<?php

 $sbsname = $_REQUEST["sbs"];                    (1)
 $result=`system "wrkactjob sbs($sbsname)" `; (2)
 echo("<pre>" );
 for($line=strtok($result,"\n"); $line;
   $line=strtok("\n")  )  {                      (3)
   echo("<br>$line") ;
 }
 echo("End of page");
 echo("</pre>" );
?>
</body>
</html>
```

In order to "deploy" the sample scripts, all you need to do is copy the relevant PHP or html files to the root directory /www/zendcore/htdocs. Point your browser correctly and the script is ready to run!

Scenario 1 – System Command

PHP can execute simple commands on the underlying operating system. This can be used for very simple integration needs.

The first coding example performs a simple command execution: WRKACTJOB. It demonstrates the use of a system command and a simple approach to pass parameters to the command.

Notes for the code sample:

(1) $REQUEST allows the PHP script to get parameters from the Web either using Get or Post. The statement will assign the variable $sbsname the value received by the http request variable sbs. In our example we call the script like this:

`http://smbt520:89/wrkactjob.php?sbs=QCMN`

(2) The script performs a system command (in our example it runs WRKACTJOB). It stores the result in variable $RESULT. The command accepts parameter $sbsname.

(3) A simple PHP loop to parse the variable $RESULT. The loop separates the $result variable into several lines.

Calling the example:

The following url can be used to call the example. (You will need to specify your server name or address instead of our server name SMBT520). You may need to set a different port depending on your local configuration.)

`http://smbt520:89/wrkactjob.php?sbs=QSYSWRK`

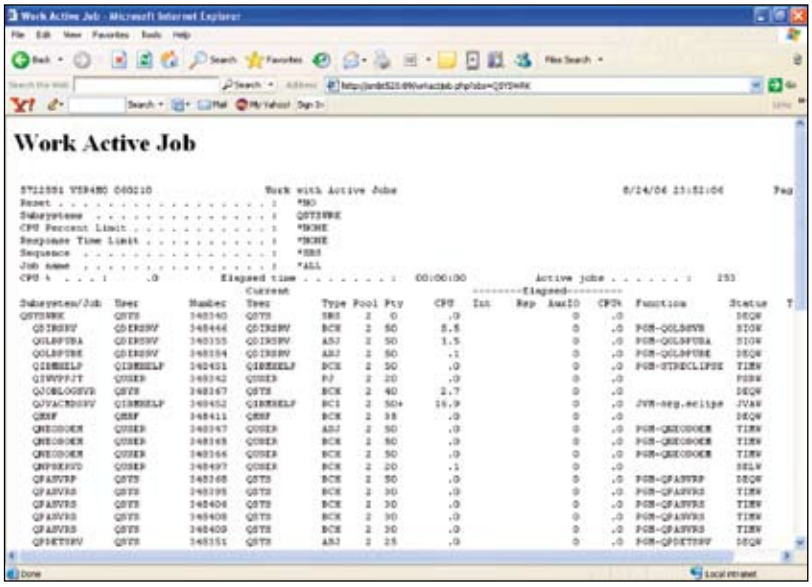

Figure 1.

Results: The screen in **Figure 1** is the result of calling WRKACTJOB.PHP with parameter sbs = QSYSWRK.

Scenario 2 – Get Records From the Database

PHP is delivered with an extension to allow access to DB2. The extension contains many features and functions and needs no special setup to work with. The second code sample performs a very basic query to an iSeries table.

Notes for the code sample:

(1) Variable $sql contains an SQL request to perform. We could easily make this statement dynamic based on user input (as previously described on the command example.)

Sample Code 2:

```
<head><title>Data base</title></head>
<body>
<?php
 $sql = "SELECT CUSNUM, LSTNAM, INIT,
 STREET FROM qiws.qcustcdt" ;                    (1)
 $con = db2_connect("SMBT520","uuuu","pwd") ; (2)
 if(!$con) {
   print("Error on connect");
   return -1;
 }
 $res = db2_exec($con, $sql ) ;                  (3)
 if(!$res) {
   print("Error on execute");
   return -1;
 }
 echo("<table border=1 bgcolor=yellow>");
 while( $row = db2_fetch_array($res) ) {         (4)
   echo("<tr>");
   echo("<td>$row[0]</td><td>$row[1]</td>
   <td>$row[2]</td>
   <td>$row[3]</td>" );
   echo("</tr>");
 }
 echo("</table>");
 echo("End of page");
?>
```

(2) db2_connect is used to connect to the database. The first parameter is the database name that can be viewed by issuing the WRKRDBDIRE command. The second and third parameters contain a valid user name and password.

(3) The statement executes the SQL request to the connection object.

(4) db2_fetch_array can be used to extract rows from the result set.

All other statements are meant to build the user interface — an ugly yellow table in our example.

Figure 2.

Calling the example: The following url can be used to call the example. (You will need to specify your server name or address instead of our server name SMBT520, and you may need to set a different port depending on your local configuration.)

`http://smbt520:89/DataBase.php`

Results: Figure 2 shows the result of calling the database.

Scenario 3 – Call an Existing Program with Return Parameters

PHP toolkit is an extension that allows access to System i resources. It is similar in functionality to the java toolbox and can be used (among other things) to call back-end programs and pass input and output parameters. The third code set is used to call an iSeries program with three parameters.

Notes for the code sample:

(1) i5_connect is used to connet to the required server. In our example we connect to the local machine by using user *uuu* and password *pwd*.

(2) Preperation of three parameters to be used on the program call. All parameters are used for both input and output.

(3) i5_program_prepare prepares the program call to program MYPGM located in MYLIB. The program is prepared with the parameters array.

(4) i5_program_call is used to perform the program call. It calls MYLIB/MYPGM with the required parameters.

(5) In our example we expect the program MYPGM to return the third parameter as a concatenation of the first 2 parameters. (If not we raise an error message.)

Calling the example: The following url can be used to call the example (You will need to specify your server name or address instead of our server name SMBT520). You may need to set a different port depending on your local configuration):

`http://smbt520:89/ProgramCall.php`

Results: The screen in **Figure 3** is the result of calling the program call example.

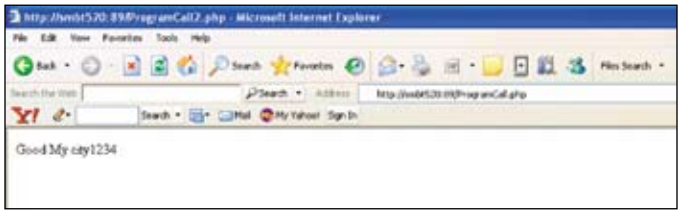

Figure 3.

Consclusion

The possibilities to integrate PHP scripts with System i resources were improved and it is now possible to develop PHP scripts with tight integration. The addition of the PHP toolkit in my opinion makes PHP an excellent option with which to Web-enable iSeries applications.

Proposed Resources

The PHP on i5 forum http://www.zend.com/forums/ contains valuable advise and examples. For example I found these two database application samples:

1.) Simple registration form: http://www.zend.com/forums/index.php?t=msg&th=1518&start=0&S=b172818554dee4e61096f1ec428c0e8e

2.) Simple file browser: http://www.zend.com/forums/index.php?t=msg&th=1368&start=0&S=b172818554dee4e61096f1ec428c0e8e

Sample Code 3:

```
<?php
/* Connect to i5 Machine */
$conn = i5_connect("127.0.0.1", "uuu", "pwd");                    (1)
if ($conn === false) {
 die("FAIL : Failed to connect to server : $i5_server_ip,
with user name : $i5_uname and password : $i5_pass");
}


/* Prepare File for execution */                                  (2)
$desc = array (
 array ("Name"=>"city", "IO"=>I5_INOUT,
     "Type" => I5_TYPE_CHAR, "Length" => "15"),
 array ("Name"=>"zip", "IO"=>I5_INOUT,
     "Type" => I5_TYPE_CHAR, "Length" => "5"),
 array ("Name"=>"result", "IO"=>I5_INOUT,
     "Type" => I5_TYPE_CHAR, "Length" => "40")
    );

$prog = i5_program_prepare("MYLIB/MYPGM", $desc);                 (3)
if ($prog === FALSE){
 $errorTab = i5_error();
 echo 'Program prepare failed \n';
var_dump($errorTab);
 die();
}


/* Execute Program */
$params = array ("city" => "My city",
     "zip"  => "1234");
$retvals = array("result" => "result");
$ret = i5_program_call($prog,$params, $retvals) ;                 (4)
if ($ret === FALSE){
$errorTab = i5_error();
 echo 'FAIL : i5_program_call failure code \n';
 var_dump($errorTab);
 die();
}


if ($result != $params[city].$params[zip])  {                     (5)
 print ("FAIL : The program returned {$result}
 but it should have been {$params[city]}{$params[zip]}");
} else {
 print ("Good {$result}\n");
}
i5_close($conn) || print
("FAIL : Failed to disconnect from server : $i5_server_ip");

?>
```

Shahar Mor *is director of iSeries services at MidLink Canada (www.midlinkca.com). A Certified iSeries Specialist with 17 years of experience working in the iSeries / AS400 / S38 environments, Mr. Mor has extensive knowledge in the fields of iSeries databases, connectivity, performance, and security. He previously held management and senior development positions in Migdal Insurance, Bank Investec and Elite industries. Shahar has written a Redbook for IBM on iSeries e-commerce (http://www.redbooks.ibm.com/redbooks/pdfs/sg245198.pdf) and is a Search400.com site expert for connectivity issues. Shahar was responsible for the porting efforts performed by ZEND to bring PHP to System i5. Shahar can be reached at mailto: shahar@midlinkca.com*